PATENT
Docket No. 0026-0030

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re Patent Application of	)	
	)	
Sanjay GHEMAWAT et al.	)	**ATTN: APPEAL BRIEF - PATENTS**
	)	
Application No.: 10/608,039	)	Group Art Unit: 2161
	)	
Filed: June 30, 2003	)	Examiner: C. Daye
	)	
For: GARBAGE COLLECTING SYSTEMS AND METHODS	)	
	)	

U.S. Patent and Trademark Office
Customer Window, Mail Stop Appeal Brief - Patents
Randolph Building
401 Dulany Street
Alexandria, VA 22314

APPEAL BRIEF

This Appeal Brief is submitted in response to the final Office Action, dated June 14, 2006, and in support of the Notice of Appeal, filed October 13, 2006.

I. REAL PARTY IN INTEREST

The real party in interest in this appeal is Google Inc.

II. RELATED APPEALS, INTERFERENCES, AND JUDICIAL PROCEEDINGS

Appellants are unaware of any related appeals, interferences, or judicial proceedings.

III. STATUS OF CLAIMS

Claims 1-25 are pending in this application.

Claims 1, 2, 4-7, 10-13, and 20-25 have been finally rejected under 35 U.S.C. § 102(b) as anticipated by Mattis et al. (U.S. Patent No. 6,209,003).

Claim 3 has been finally rejected under 35 U.S.C. § 103(a) as unpatentable over Mattis et al. in view of Manley et al. (U.S. Patent Application Publication No. 2003/0182330).

Claims 8, 9, and 14-19 have been finally rejected under 35 U.S.C. § 103(a) as unpatentable over Mattis et al. in view of Hisgen et al. ("New-Value Logging in the Echo Replicated File System," June 23, 1993).

Claims 1-25 are the subject of the present appeal. These claims are reproduced in the Claim Appendix of this Appeal Brief.

IV. STATUS OF AMENDMENTS

An After Final Request for Reconsideration was filed on August 14, 2006, subsequent to the final Office Action. No claim amendments were filed subsequent to the final Office Action. The Examiner issued an Advisory Action, dated September 11, 2006, that indicated that the After Final Request for Reconsideration did not place the application in condition for allowance. A Pre-Appeal Brief Request for Review was filed on October 13, 2006, which resulted in dismissal for allegedly including more than five pages.

V. SUMMARY OF CLAIMED SUBJECT MATTER

In the paragraphs that follow, a concise explanation of the independent claims and the claims reciting means-plus-function or step-plus-function language that are involved in this

appeal will be provided by referring, in parenthesis, to examples of where support can be found in the specification and drawings.

Claim 1 recites a method for deleting one or more of a plurality of files (Fig. 9), the files including one or more chunks stored by a plurality of servers (Fig. 1, 120; paragraph 0031). The method comprises identifying a file to be deleted (paragraph 0071); renaming the identified file (paragraph 0071); permanently deleting the renamed file a predetermined amount of time after renaming the identified file as part of a garbage collection process (paragraph 0072); receiving, from the servers, information concerning chunks stored by the servers (paragraph 0072); and identifying, to one of the servers, one of the chunks that corresponds to the permanently deleted file (paragraph 0072).

Claim 12 recites a system (Fig. 1, FILE SYSTEM) for deleting a file that includes data stored by a plurality of servers (Fig. 1, 120; paragraph 0031). The system comprises means for identifying a file to be deleted (paragraph 0071); means for logging deletion of the identified file (paragraph 0071); means for permanently deleting the file during a garbage collection process that occurs after logging deletion of the identified file (paragraph 0072); means for receiving, from the servers, information concerning data stored by the servers (paragraph 0072); and means for identifying, to one of the servers, that of the data that corresponds to the file that was permanently deleted (paragraph 0072).

Claim 13 recites a file system (Fig. 1, FILE SYSTEM) that comprises a plurality of servers (Fig. 1, 120) configured to store files as chunks (paragraph 0031), each of the files including one or more of the chunks (paragraph 0031); and a master (Fig. 1, 130) connected to the servers (Fig. 1, 120) and configured to identify one of the files to be deleted (paragraph

0071), rename the identified file (paragraph 0071), permanently delete one or more chunks associated with the renamed file a predetermined amount of time after renaming the identified file as part of a garbage collection process (paragraph 0072), receive, from the servers (Fig. 1, 120), information concerning chunks stored by the servers (Fig. 1, 120; paragraph 0072), and identify, to one of the servers (Fig. 1, 120), one of the chunks that corresponds to one of the one or more permanently deleted chunks (paragraph 0072).

Claim 14 recites a method for deleting orphaned chunks of a plurality of chunks stored by a plurality of servers (Fig. 1, 120; Fig. 9). The method comprises providing a mapping of file names to chunks (paragraphs 0041-0042); identifying chunks, as orphaned chunks, that are not reachable from any of the file names (paragraph 0073); deleting the orphaned chunks (paragraph 0073); receiving, from the servers, information concerning chunks stored by the servers (paragraph 0074); and identifying, to one of the servers, one of the chunks that corresponds to one of the deleted orphaned chunks (paragraph 0074).

Claim 18 recites a system for deleting orphaned chunks of a plurality of chunks stored by a plurality of servers (Fig. 1, 120; Fig. 9; paragraph 0031; paragraph 0073). The system comprises means for mapping file names to chunks (paragraphs 0041-0042); means for identifying chunks, as orphaned chunks, that are not reachable from any of the file names (paragraph 0073); means for deleting the orphaned chunks as part of a garbage collection process (paragraph 0073); means for receiving, from the servers, information concerning chunks stored by the servers (paragraph 0074); and means for identifying, to one of the servers, one of the chunks that corresponds to one of the deleted orphaned chunks (paragraph 0074).

Claim 19 recites a file system (Fig. 1, FILE SYSTEM) that comprises a plurality of

servers (Fig. 1, 120) configured to store files as chunks (paragraph 0031), each of the files including one or more of the chunks (paragraph 0031); and a master (Fig. 1, 130) connected to the servers (Fig. 1, 120) and configured to map file names to chunks (paragraphs 0040-0042), identify chunks, as orphaned chunks, that are not reachable from any of the file names (paragraph 0073), delete the orphaned chunks (paragraph 0073), receive, from the servers, information concerning chunks stored by the servers (paragraph 0074), and identify, to one of the servers, one of the chunks that corresponds to one of the deleted orphaned chunks (paragraph 0074).

Claim 20 recites a method for deleting stale replicas of chunks (Fig. 9), the replicas being stored by a plurality of servers (Fig. 1, 120, paragraph 0031). The method comprises associating version information with replicas of chunks (paragraphs 0042-0043; paragraphs 0075-0076); identifying stale replicas based on the associated version information (paragraph 0076); deleting the stale replicas (paragraph 0077; paragraph 0073); receiving, from the servers, information concerning replicas stored by the servers (paragraph 0077; paragraph 0074); and identifying, to one of the servers, one of the replicas that corresponds to one of the deleted stale replicas (paragraph 0077; paragraph 0074).

Claim 24 recites a system for deleting stale replicas of chunks, the replicas being stored by a plurality of servers (Fig. 1, 120; paragraph 0031). The system comprises means for generating version information for replicas of chunks (paragraphs 0042-0043; paragraphs 0075-0076); means for identifying stale replicas based on the generated version information (paragraph 0076); means for deleting the stale replicas as part of a garbage collection process (paragraph 0077; paragraph 0073); means for receiving, from the servers, information

concerning replicas stored by the servers (paragraph 0077; paragraph 0074); and means for identifying, to one of the servers, one of the replicas that corresponds to one of the deleted stale replicas (paragraph 0077; paragraph 0074).

Claim 25 recites a file system (Fig. 1, FILE SYSTEM) that stores files as chunks (paragraph 0031). The file system (Fig. 1, FILE SYSTEM) comprises a plurality of servers (Fig. 1, 120) configured to store files as chunks (paragraph 0031); and a master (Fig. 1, 130) connected to the servers (Fig. 1, 120) and configured to associate version information with the chunks (paragraphs 0042-0043; paragraphs 0075-0076), identify stale chunks based on the associated version information (paragraph 0076), delete the stale chunks (paragraph 0077; paragraph 0073), receive, from the servers, information concerning replicas stored by the servers (paragraph 0077; paragraph 0074), and identify, to one of the servers, one of the replicas that corresponds to one of the deleted stale chunks (paragraph 0077; paragraph 0074).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. Claims 1, 2, 4-7, 10-13, and 20-25 stand rejected under 35 U.S.C. § 102(b) as anticipated by Mattis et al.

B. Claim 3 stands rejected under 35 U.S.C. § 103(a) as unpatentable over Mattis et al. in view of Manley et al.

C. Claims 8, 9, and 14-19 stand rejected under 35 U.S.C. § 103(a) as unpatentable over Mattis et al. in view of Hisgen et al.

VII. ARGUMENT

A. The Rejection Under 35 U.S.C. § 102(b) Based on Mattis et al. (U.S. Patent No. 6,209,003) Should be Reversed.

The initial burden of establishing a prima facie basis to deny patentability to a claimed invention is always upon the Examiner. In re Oetiker, 977 F.2d 1443, 24 USPQ2d 1443 (Fed. Cir. 1992). For a proper rejection under 35 U.S.C. § 102, each and every element as set forth in the claim must be found in a single prior art reference. Verdegaal Bros. v. Union Oil Co. of California, 814 F.2d 628, 2 USPQ2d 1051 (Fed. Cir. 1987). Prior legal precedent requires that the identical invention be shown in as complete detail as is contained in the claim. Richardson v. Suzuki Motor Co., 868 F.2d 1226, 9 USPQ2d 1913 (Fed. Cir. 1989).

1. Claims 1, 2, 6, and 7.

Independent claim 1 is directed to a method for deleting one or more of a plurality of files, where the files include one or more chunks stored by a plurality of servers. The method comprises identifying a file to be deleted; renaming the identified file; permanently deleting the renamed file a predetermined amount of time after renaming the identified file as part of a garbage collection process; receiving, from the servers, information concerning chunks stored by the servers; and identifying, to one of the servers, one of the chunks that corresponds to the permanently deleted file.

Mattis et al. does not disclose or suggest the combination of features recited in claim 1. For example, Mattis et al. does not disclose or suggest renaming a file that is identified to be deleted. The Examiner alleged that Mattis et al. discloses this feature and cited column 3, lines 16-19, of Mattis et al. for support. Final Office Action, page 3. Appellants disagree.

At column 3, lines 13-19, Mattis et al. discloses:

Also, file systems, being designed for user data file management, include facilities

> irrelevant to cache object stores, and indeed counter-productive to this application. Examples include: support for random access and selective modification, file permissions, support for moving files, support for renaming files, and support for appending to files over time.

This section of Mattis et al. discloses that file systems, which include a prior art approach to structure cache object stores, include support for renaming files. This portion specifically states that renaming files is "irrelevant to cache object stores" and "counter-productive to [the Mattis et al.] application." Column 3, lines 13-19. In other words, this section of Mattis et al. specifically teaches away from renaming files. Instead, Mattis et al. discloses that if a fragment is to be deleted, then it is deleted by marking it as deleted and overwriting the data in the fragment. Column 23, lines 15-17. Nowhere does Mattis et al. disclose or suggest renaming a file that is identified to be deleted, as required by claim 1.

Nevertheless, this section of Mattis et al. discloses a prior art approach that supports renaming files. Contrary to the Examiner's allegation, however, this prior art approach described by Mattis et al. does not disclose or remotely suggest renaming a file that is identified to be deleted, as required by claim 1.

Further, even if, for the sake of argument, this prior art approach described by Mattis et al. could be equated to renaming a file that is identified to be deleted (a point that Appellants submit is unreasonable for the reasons provided herein), the Examiner has identified a prior art approach as allegedly equivalent to this feature of claim 1 but relied upon the system of Mattis et al. for other features of claim 1. As noted above, the Mattis et al. system does not include this prior art approach and Mattis et al. specifically teaches away from including this prior art approach, and the Examiner has not provided the requisite motivation for modifying the Mattis et al. system to include this prior art approach. Accordingly, the Examiner has not established a

proper rejection under 35 U.S.C. § 102 with regard to claim 1.

The Examiner also alleged that Mattis et al. discloses:

> a method for garbage collection, which means it is a method for detecting and freeing (i.e. deletion) of unwanted memory. Therefore, since the sole of the invention is to identify space that needs to be deleted, it is common practice that the renaming of files would be assigned to the files, which were identified as being deleted. To further elaborate, Mattis discloses at column 23, lines 15-23, wherein if a fragment is to be deleted, the fragment is marked as deleted, which means that the fragment is an altered fragment that has now been renamed because of the deletion status. As a result, Mattis discloses renaming a file that is identified to be deleted.

Final Office Action, page 10. The Examiner's allegation does not find support in the disclosure of Mattis et al. The Examiner alleged that "it is common practice that the renaming of files would be assigned to the files" (emphasis added) (final Office Action, page 10), but the Examiner has not identified any portion of the disclosure of Mattis et al. or any other evidence that supports the Examiner's allegation.

At column 23, lines 15-23, Mattis et al. discloses:

> If the fragment is to be deleted, then in step 812 it is deleted from the arena by marking it as deleted and overwriting the data in the fragment. When an object 52 is stored in multiple fragments, and the garbage collection process determines that one of the fragments is to be deleted, then the process deletes all fragments associated with the object. This may involve following a chain of fragments, of the type shown in FIG. 5, to another arena or even another pool.

In this section, Mattis et al. discloses that if a fragment is to be deleted, it is deleted by marking it as deleted and overwriting the data in the fragment. Nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest renaming a file that is identified to be deleted, as required by claim 1.

Mattis et al. also does not disclose or suggest permanently deleting the renamed file a predetermined amount of time after renaming the identified file as part of a garbage collection process, as further recited in claim 1. The Examiner alleged that Mattis et al. discloses this

feature and cited column 16, lines 48-52; column 21, lines 52-55; and column 22, lines 43-47, of Mattis et al. for support. Final Office Action, page 3. Appellants disagree.

At column 16, lines 48-52, Mattis et al. discloses:

> For example, the Open Directory is useful in safeguarding against overwriting or deleting an object that is currently being read. The Open Directory also buffers changes to the Directory Table 110 before they are given permanent effect in the Directory Table 110.

In this section, Mattis et al. discloses that the Open Directory buffers changes to the Directory Table before they are given permanent effect in order to safeguard against overwriting or deleting an object that is currently being read. Nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest permanently deleting a renamed file a predetermined amount of time after renaming a file that is identified to be deleted as part of a garbage collection process, as required by claim 1.

The Examiner alleged that this section of Mattis et al. means that the act of deleting is buffered to make sure that it is the correct action that needs to be taken and, if so, the deletion is made permanent. Final Office Action, page 11. Appellants submit that the Examiner is misinterpreting this section of Mattis et al. This section states that changes to the Directory Table are buffered to safeguard against overwriting or deleting an object that is currently being read. This section does not, as alleged by the Examiner, state that a deletion is buffered to make sure that it is the correct action that needs to be taken. Therefore, the Examiner's interpretation of this section of Mattis et al. is flawed.

At column 21, lines 52-55, Mattis et al. discloses:

> Preferably, the garbage collection method is implemented as an independent process that runs in parallel with other processes that relate to the cache. This enables the garbage collection method to periodically clean up cache storage areas without interrupting or affecting the operation of the cache.

In this section, Mattis et al. discloses that a garbage collection method periodically cleans up the cache storage areas. Nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest permanently deleting a renamed file a predetermined amount of time after renaming a file that is identified to be deleted as part of a garbage collection process, as required by claim 1.

At column 22, lines 39-47, Mattis et al. discloses:

> In step 806, one of the fragments within the selected arena is selected for garbage collection. In determining which fragment or fragments to select, the cache 80 takes into account several selection factors, as indicated by block 807. In the preferred embodiment, the factors include: the time of the last access to the fragment; the number of hits that have occurred to an object that has data in the fragment; the time required to download data from the fragment to a client; and the size of the object of which the fragment is a part.

In this section, Mattis et al. discloses factors that are taken into consideration when determining which fragment to select for garbage collection. Nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest permanently deleting a renamed file a predetermined amount of time after renaming a file that is identified to be deleted as part of a garbage collection process, as required by claim 1.

The Examiner alleged that the factors that the Mattis et al. system considers when determining which fragment to select for garbage collection "demonstrate a predetermined amount of time after renaming." Final Office Action, page 11. Appellants submit that the Examiner's conclusion finds no support in the disclosure of Mattis et al. and falls short of establishing a proper rejection under 35 U.S.C. § 102.

The Examiner also alleged that:

> [t]he limitation of "permanently deleting a file" has not been entirely disclosed within [Appellants'] specification, since even though a user may mark a file as deleted and proceed to further remove the file, it is known that the "deleted/removed file" is truly not permanently deleted, because the information is still within the computer system and if

> needed, could possibly be retrieved

(emphasis in original). Final Office Action, pages 10-11. Appellants submit that Appellants' original specification provides full support for the claimed features. For example, support for the claim feature of "permanently deleting the renamed file . . . " can be found in the specification at paragraph 0072. Therefore, the Examiner's allegation that this feature is not disclosed in the specification is without merit.

The Examiner further cited column 32, lines 29-37, of Mattis et al. for allegedly disclosing that a block is set with a deletion flag indicating that the block will ultimately be deleted and eventually, the block is removed from the directory, thereby permanently deleting the file. Final Office Action, page 11. At column 32, lines 29-37, Mattis et al. discloses:

> To accomplish removal of a block found in the cache, however, in step 960 the process sets the deletion flag, and checks the block in with the deletion flag set. As described herein in connection with the check-in process (steps 938 and 944 of FIG. 9B), when the deletion flag is set, the block will be marked as deleted. Thereafter, the block is eventually removed from the Directory Index when the changes reflected in the Open Directory are synchronized to the Directory Index.

In this section, Mattis et al. discloses that to remove a block, a deletion flag is set, the block is marked as deleted, and the block is eventually removed from the Directory Index. Contrary to the Examiner's allegation, nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest permanently deleting a renamed file a predetermined amount of time after renaming a file that is identified to be deleted as part of a garbage collection process, as required by claim 1.

The Examiner further cited column 22, lines 14-23, of Mattis et al. for allegedly disclosing:

> high and low water marks need to be at a certain value, and if the water marks are not at those values the garbage collection is carried out at a time before the capacity is exceeded. This example not only discloses a predetermined amount of time, but it also discloses the fact this is all part of a garbage collection process. As a result, Mattis

> discloses permanently deleting the renamed file a predetermined amount of time after renaming the identified file as part of a garbage collection process.

Final Office Action, page 11. Appellants submit that the Examiner's conclusion finds no support in the disclosure of Mattis et al. and results solely from the use of impermissible hindsight reasoning.

At column 22, lines 14-23, Mattis et al. discloses:

> When the amount of active storage in a particular pool becomes greater than the "high water mark" value, garbage collection is initiated and carried out repeatedly until the amount of active storage in the pool falls below the "low water mark" value. The "low water mark" value is selected to be greater than zero, and the "high water mark" value is chosen to be approximately 20% less than the total storage capacity of the pool. In this way, garbage collection is carried out at a time before the pool overflows or the capacity of the storage device 90a is exceeded.

In this section, Mattis et al. discloses that garbage collection is initiated when the amount of active storage in the pool becomes greater than a high water mark value and continues until the amount of active storage in the pool falls below a low water mark value. Nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest permanently deleting a renamed file a predetermined amount of time after renaming a file that is identified to be deleted as part of a garbage collection process, as required by claim 1.

In the Advisory Action, the Examiner also cited column 22, lines 14-47, of Mattis et al. for allegedly disclosing permanently deleting a renamed file a predetermined amount of time after renaming a file that is identified to be deleted as part of a garbage collection process. Advisory Action, page 2. Appellants disagree.

Column 22, lines 14-23 and 39-47, of Mattis et al. are reproduced and discussed above. At column 22, lines 24-38, Mattis et al. discloses:

> 2. USAGE-AWARE GARBAGE COLLECTION

> In step 804, one of the arenas is selected as a target for carrying out garbage collection. The arena is selected by a selection algorithm that considers various factors. As indicated by block 805, the factors include, for example, whether the arena is the last arena accessed by the cache 80, and the total number of accesses to the arena. In alternate embodiments, the factors may also include the number of information objects that have been deleted from each arena, how recently an arena has been used, how recently garbage collection was previously carried out on each arena, and whether an arena currently has read or write locks set on it. Once the arena is selected for garbage collection, all of the fragments inside the object are separately considered for garbage collection.

In this section, Mattis et al. discloses selecting an arena for garbage collection based on various factors. Nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest permanently deleting a renamed file a predetermined amount of time after renaming a file that is identified to be deleted as part of a garbage collection process, as required by claim 1.

Claim 1 recites additional features that are neither disclosed nor suggested by Mattis et al. For example, Mattis et al. does not disclose or suggest identifying, to one of the servers, one of the chunks that corresponds to the permanently deleted file, as further recited in claim 1. The Examiner alleged that Mattis et al. discloses this feature and cited column 33, lines 32-40, of Mattis et al. for support. Final Office Action, page 3. Appellants disagree.

At column 33, lines 32-40, Mattis et al. discloses:

> If no match of the key is found in the search, then in step 848 the process returns an error message to the calling program or process, indicating that the requested object does not exist in the cache. Although the specific response to such a message is determined by the calling program or process, in the World Wide Web context, generally the proxy 30 contacts the server 40 that stores the object using an HTTP request, and obtains a copy of the requested object.

In this section, Mattis et al. discloses that a search is performed for a key to determine whether a requested object exists in the cache and if no match of the key is found, the proxy contacts the server to obtain a copy of the requested object. Nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest identifying, to one of the servers, one of the chunks that

corresponds to a file that was permanently deleted a predetermined amount of time after renaming the file, as required by claim 1.

Relying on this section of Mattis et al., the Examiner states:

> [t]he requested object is known as being permanently deleted because initially the search receives an error message stating the object does not exist. If the object does not exist it is obvious that the object was permanently deleted. Therefore, Mattis does disclose identifying, to one of the servers, one of the chunks that corresponds to the permanently deleted file.

Final Office Action, page 12. Appellants submit that the Examiner does not understand the basic operation of a cache. When a cache miss occurs (i.e., when a request for an object from the cache is made and the object is not present in the cache), this does not necessarily mean that object was once present in the cache and subsequently permanently deleted. Instead, it simply means that the object is not currently stored by the cache. For example, it is possible that the object was never stored by the cache. Thus, the Examiner's allegation is flawed and the Examiner's conclusion, which is based on this erroneous assumption, is equally flawed and finds no support in the disclosure of Mattis et al.

For at least these reasons, it is respectfully submitted that claims 1, 2, 6, and 7 are not anticipated by Mattis et al. under 35 U.S.C. § 102. Reversal of the rejection of claims 1, 2, 6, and 7 is respectfully requested.

2. Claim 4.

Dependent claim 4 recites that the predetermined amount of time is a user-configurable amount of time.

Initially, claim 4 depends from claim 1. Claim 4 is, therefore, not anticipated by Mattis et al. for at least the reasons given with regard to claim 1.

Additionally, Mattis et al. does not disclose or suggest the combination of features recited in claim 4. As explained above with regard to claim 1, Mattis et al. is completely silent with regard to permanently deleting a renamed file a predetermined amount of time after renaming the file. Therefore, Mattis et al. cannot disclose or suggest that the predetermined amount of time is a user-configurable amount of time, as required by claim 4. In other words, Mattis et al. is completely silent with regard to permanently deleting a renamed file a user-configurable amount of time after renaming the file, as required by claim 4.

The Examiner alleged that Mattis et al. discloses this feature and cited column 22, lines 14-23, of Mattis et al. for support. Final Office Action, page 3. Appellants disagree.

Column 22, lines 14-23, of Mattis et al. is reproduced above. In this section, Mattis et al. discloses that garbage collection is initiated when the amount of active storage in a pool becomes greater than a high water mark. Nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest permanently deleting a renamed file a user-configurable amount of time after renaming the file, as required by claim 4.

The Examiner alleged that the high and low water marks in this section of Mattis et al.

> were assigned within the application during the programming stages, which is predetermined and done by the user (i.e. programmer). Therefore, Mattis does disclose the predetermined amount of time is a user-configurable amount of time.

Final Office Action, pages 12-13. The Examiner's allegation finds no basis in fact. As explained above with regard to claim 1, Mattis et al. does not disclose or suggest permanently deleting a renamed file a predetermined amount of time after renaming the file. Therefore, even assuming, for the sake of argument, that a programmer of an application in Mattis et al. can be considered a user (a point that Appellants do not concede), Mattis et al. does not disclose or remotely suggest

permanently deleting a renamed file a user-configurable amount of time after renaming the file, as required by claim 4. The Examiner's allegations to the contrary find no support in the disclosure of Mattis et al. and result solely from the use of impermissible hindsight reasoning.

In the Advisory Action, the Examiner cited column 22, lines 8-23, of Mattis et al. for allegedly disclosing a user-configurable amount of time. Advisory Action, page 2. Appellants disagree.

Column 22, lines 14-23, has been reproduced and discussed above. At column 22, lines 8-14, Mattis et al. discloses:

> In step 802, one of the pools 200a-200n is selected for garbage collection operations. Preferably, for each pool 200a-200n of a storage device 90a, the cache stores or can access a value indicating the amount of disk space in a pool that is currently storing active data. The cache also stores constant "low water mark" and "high water mark" values, as indicated by block 803.

In this section, Mattis et al. discloses selecting a pool for garbage collection. Nowhere in this section, or elsewhere, does Mattis et al. disclose or remotely suggest permanently deleting a renamed file a user-configurable amount of time after renaming the file, as required by claim 4.

For at least these reasons, it is respectfully submitted that claim 4 is not anticipated by Mattis et al. under 35 U.S.C. § 102. Reversal of the rejection of claim 4 is respectfully requested.

3. Claim 5.

Dependent claim 5 recites that the user-configurable amount of time differs for different ones of the files.

Initially, claim 5 depends from claim 4. Claim 5 is, therefore, not anticipated by Mattis et al. for at least the reasons given with regard to claim 4.

Additionally, Mattis et al. does not disclose or suggest the combination of features recited in claim 5. As explained above with regard to claim 4, Mattis et al. is completely silent with regard to permanently deleting a renamed file a user-configurable amount of time after renaming the file. Therefore, Mattis et al. cannot disclose or suggest that the user-configurable amount of time differs for different ones of the files, as required by claim 5. In other words, Mattis et al. is completely silent with regard to permanently deleting a renamed file a user-configurable amount of time after renaming the file, where the user-configurable amount of time differs for different ones of the files, as required by claim 5.

The Examiner alleged that Mattis et al. discloses this feature and cited column 22, line 65 - column 23, line 5, of Mattis et al. for support. Final Office Action, page 4. Appellants disagree.

At column 22, line 65 - column 23, line 5, Mattis et al. discloses:

> By storing values that identify the time required to download an object, the size of the object, and the number of times the object was hit in cache, the garbage collector can estimate, for each object, how much server download time was avoided and how much server traffic was disabled, by serving the cached copy as opposed to fetching from the original server. This metric measures the inherent "value" of the cached object.

In this section, Mattis et al. discloses that the value of a cached object can be estimated by how much server download time was avoided and how much server traffic was disabled. Nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest permanently deleting a renamed file a user-configurable amount of time after renaming the file, where the user-configurable amount of time differs for different ones of the files, as required by claim 5.

For at least these reasons, it is respectfully submitted that claim 5 is not anticipated by Mattis et al. under 35 U.S.C. § 102. Reversal of the rejection of claim 5 is respectfully requested.

4. Claims 10 and 11.

Dependent claim 10 recites maintaining versions of the chunks; identifying a stale chunk based on the versions of the chunks; and deleting the stale chunk.

Initially, claim 10 depends from claim 1. Claim 10 is, therefore, not anticipated by Mattis et al. for at least the reasons given with regard to claim 1.

Additionally, Mattis et al. does not disclose or suggest the combination of features recited in claim 10. For example, Mattis et al. does not disclose or suggest identifying a stale chunk based on the versions of the chunks.

The Examiner alleged that Mattis et al. discloses identifying a stale chunk based on the versions of the chunks and cited column 26, lines 15-21, of Mattis et al. for support. Final Office Action, page 4. Appellants disagree.

At column 26, lines 15-22, Mattis et al. discloses:

> In block 1216, the cache updates an expiration date value stored in association with the information object to reflect the current date or time. By updating the expiration date, the cache ensures that the garbage collection process will not delete the object, because after the update it is not considered old. In this way, an old object is refreshed in the cache without retrieving the object from its origin, writing it in the cache, and deleting a stale copy of the object.

In this section, Mattis et al. discloses that the cache updates an expiration date value stored in association with an information object. Appellants submit that nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest identifying a stale chunk based on the versions of the chunks, as required by claim 10.

Even assuming, for the sake of argument, that this section of Mattis et al. can reasonably be understood to disclose identifying a stale chunk (a point that Appellants do not concede), Appellants submit that the Examiner's rejection is flawed. The Examiner alleged that the version

number of the program or process that created an arena in a pool (column 17, lines 42-46) is allegedly equivalent to a version of a chunk. Final Office Action, page 4. With this interpretation in mind, nowhere in column 26, lines 15-21, does Mattis et al. disclose or remotely suggest identifying a stale chunk based on a version number of a program or process that created an arena in a pool. Therefore, column 26, lines 15-21, of Mattis et al. cannot disclose or suggest identifying a stale chunk based on the versions of the chunks, as required by claim 10.

For at least these reasons, it is respectfully submitted that claims 10 and 11 are not anticipated by Mattis et al. under 35 U.S.C. § 102. Reversal of the rejection of claims 10 and 11 is respectfully requested.

5. Claim 12.

Independent claim 12 is directed to a system for deleting a file that includes data stored by a plurality of servers. The system comprises means for identifying a file to be deleted; means for logging deletion of the identified file; means for permanently deleting the file during a garbage collection process that occurs after logging deletion of the identified file; means for receiving, from the servers, information concerning data stored by the servers; and means for identifying, to one of the servers, that of the data that corresponds to the file that was permanently deleted.

Mattis et al. does not disclose or suggest the combination of features recited in claim 12. For example, Mattis et al. does not disclose or suggest means for permanently deleting a file during a garbage collection process that occurs after logging deletion of the file. The Examiner did not specifically address this feature of claim 12, but instead rejected claim 12 by generally referring to the rejection of claim 1. Final Office Action, pages 2-3. This feature is not recited in claim 1, however. Therefore, the Examiner did not establish a proper case of anticipation with

regard to claim 12.

Mattis et al. also does not disclose or suggest means for identifying, to one of the servers, that of the data that corresponds to the file that was permanently deleted, as further recited in claim 12. The Examiner alleged that Mattis et al. discloses this feature and cited column 33, lines 32-40, of Mattis et al. for support. Final Office Action, page 3. Appellants disagree for at least reasons similar to reasons given with regard to claim 1.

For at least these reasons, it is respectfully submitted that claim 12 is not anticipated by Mattis et al. under 35 U.S.C. § 102. Reversal of the rejection of claim 12 is respectfully requested.

6. Claim 13.

Independent claim 13 is directed to a file system that comprises a plurality of servers configured to store files as chunks, each of the files including one or more of the chunks; and a master connected to the servers and configured to identify one of the files to be deleted, rename the identified file, permanently delete one or more chunks associated with the renamed file a predetermined amount of time after renaming the identified file as part of a garbage collection process, receive, from the servers, information concerning chunks stored by the servers, and identify, to one of the servers, one of the chunks that corresponds to one of the one or more permanently deleted chunks.

Mattis et al. does not disclose or suggest the combination of features recited in claim 13. For example, Mattis et al. does not disclose or suggest a master configured to rename a file that is identified to be deleted. The Examiner alleged that Mattis et al. discloses renaming a file that is identified to be deleted and cited column 3, lines 16-19, of Mattis et al. for support. Final

Office Action, page 3. Appellants disagree and submit that Mattis et al. actually teaches away from this feature for at least reasons similar to reasons given with regard to claim 1.

Mattis et al. also does not disclose or suggest a master configured to permanently delete one or more chunks associated with the renamed file a predetermined amount of time after renaming the identified file as part of a garbage collection process. The Examiner did not specifically address this feature, but instead alleged that Mattis et al. discloses permanently deleting a renamed file a predetermined amount of time after renaming the identified file as part of a garbage collection process and cited column 16, lines 48-52; column 21, lines 52-55; and column 22, lines 43-47, of Mattis et al. for support. Final Office Action, page 3. Appellants submit that Mattis et al. does not disclose or suggest a master configured to permanently delete one or more chunks associated with the renamed file a predetermined amount of time after renaming the identified file as part of a garbage collection process, as recited in claim 13, for at least reasons similar to reasons given with regard to claim 1.

Mattis et al. also does not disclose or suggest a master configured to identify, to one of the servers, one of the chunks that corresponds to one of the one or more permanently deleted chunks. The Examiner alleged that Mattis et al. discloses this feature and cited column 33, lines 32-40, of Mattis et al. for support. Final Office Action, page 3. Appellants disagree for at least reasons similar to reasons given with regard to claim 1.

For at least these reasons, it is respectfully submitted that claim 13 is not anticipated by Mattis et al. under 35 U.S.C. § 102. Reversal of the rejection of claim 13 is respectfully requested.

7. Claims 20 and 22-25.

Independent claim 20 is directed to a method for deleting stale replicas of chunks, where the replicas are stored by a plurality of servers. The method comprises associating version information with replicas of chunks; identifying stale replicas based on the associated version information; deleting the stale replicas; receiving, from the servers, information concerning replicas stored by the servers; and identifying, to one of the servers, one of the replicas that corresponds to one of the deleted stale replicas.

Mattis et al. does not disclose or suggest the combination of features recited in claim 20. For example, Mattis et al. does not disclose or suggest identifying stale replicas based on the associated version information.

The Examiner alleged that Mattis et al. discloses identifying stale replicas based on the associated version information and identified column 26, lines 15-21, of Mattis et al. for support. Final Office Action, page 5. Appellants disagree.

Column 26, lines 15-22, of Mattis et al. is reproduced above. In this section, Mattis et al. discloses that the cache uses an expiration date value to determine whether to delete an object from the cache. Nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest, for example, identifying stale replicas based on the associated version information, as required by claim 20.

Even assuming, for the sake of argument, that this section of Mattis et al. can reasonably be understood to disclose identifying stale replicas (a point that Appellants do not concede), Appellants submit that the Examiner's rejection is flawed. The Examiner alleged that the different versions of an object (e.g., versions in English, French, and Japanese languages) (column 14, lines 29-37) is allegedly equivalent to the generated version information for replicas

of chunks. Final Office Action, page 5. With this interpretation in mind, nowhere in column 26, lines 15-21, does Mattis et al. disclose or remotely suggest identifying stale replicas based on this information regarding different versions of an object. Therefore, column 26, lines 15-21, of Mattis et al. cannot disclose or suggest identifying stale replicas based on the associated version information, as required by claim 20.

Mattis et al. also does not disclose or suggest identifying, to one of the servers, one of the replicas stored by the server that corresponds to one of the deleted stale replicas, as further recited in claim 20.

The Examiner alleged that Mattis et al. discloses identifying, to one of the servers, one of the replicas that corresponds to one of the deleted stale replicas and identified column 26, lines 15-21, of Mattis et al. for support. Final Office Action, page 5. Appellants disagree.

Column 26, lines 15-22, of Mattis et al. is reproduced above. In this section, Mattis et al. discloses that the cache uses an expiration date value to determine whether to delete an object from the cache. Nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest, for example, identifying, to one of the servers, one of the replicas stored by the server that corresponds to one of the deleted stale replicas, as required by claim 20.

For at least these reasons, it is respectfully submitted that claims 20 and 22-25 are not anticipated by Mattis et al. under 35 U.S.C. § 102. Reversal of the rejection of claims 20 and 22-25 is respectfully requested.

8. Claim 21.

Dependent claim 21 recites that the version information for one of the replicas is updated each time a lease is granted for the one of the replicas.

Initially, claim 21 depends from claim 20. Claim 21 is, therefore, not anticipated by Mattis et al. for at least the reasons given with regard to claim 20.

Additionally, Mattis et al. does not disclose or suggest the combination of features recited in claim 21. The Examiner alleged that Mattis et al. discloses version information for one of the replicas that is updated each time a lease is granted for the one of the replicas and cited column 26, lines 8-15, of Mattis et al. for support. Final Office Action, page 6. Appellants disagree.

At column 26, lines 8-15, Mattis et al. discloses:

> If the Read Counter value is high, then the information object has been loaded recently. In that case, in block 1210 the cache sends a positive response message to the requesting process. Otherwise, as indicated in block 1212, the information object has not been loaded recently. Accordingly, as shown in block 1214, the cache sends a negative responsive message to the calling program or process.

In this section, Mattis et al. discloses if the read counter value is high, the cache sends a positive response message to a requesting process; otherwise, the cache sends a negative response message to the calling program or process. Nowhere in this section, or elsewhere, does Mattis et al. disclose or suggest version information for one of the replicas that is updated each time a lease is granted for the one of the replicas, as required by claim 21.

Appellants also note that the Examiner alleged that the different versions of an object (e.g., versions in English, French, and Japanese languages) (column 14, lines 29-37) is allegedly equivalent to the generated version information for replicas of chunks. Final Office Action, page 5. With this interpretation in mind, nowhere in column 26, lines 8-15, does Mattis et al. disclose or remotely suggest that the different versions of an object are updated each time a lease is granted for the object. Therefore, column 26, lines 8-15, of Mattis et al. cannot disclose or

suggest version information for one of the replicas that is updated each time a lease is granted for the one of the replicas, as required by claim 21.

For at least these reasons, it is respectfully submitted that claim 21 is not anticipated by Mattis et al. under 35 U.S.C. § 102. Reversal of the rejection of claim 21 is respectfully requested.

B. The Rejection Under 35 U.S.C. § 103(a) Based on Mattis et al. (U.S. Patent No. 6,209,003) in View of Manley et al. (U.S. Patent Application Publication No. 2003/0182330) Should be Reversed.

The initial burden of establishing a prima facie basis to deny patentability to a claimed invention is always upon the Examiner. In re Oetiker, 977 F.2d 1443, 24 USPQ2d 1443 (Fed. Cir. 1992). In rejecting a claim under 35 U.S.C. § 103, the Examiner must provide a factual basis to support the conclusion of obviousness. In re Warner, 379 F.2d 1011, 154 USPQ 173 (CCPA 1967). Based upon the objective evidence of record, the Examiner is required to make the factual inquiries mandated by Graham v. John Deere Co., 86 S.Ct. 684, 383 U.S. 1, 148 USPQ 459 (1966). The Examiner is also required to explain how and why one having ordinary skill in the art would have been led to modify an applied reference and/or combine applied references to arrive at the claimed invention. Uniroyal, Inc. v. Rudkin-Wiley Corp., 837 F.2d 1044, 5 USPQ2d 1434 (Fed. Cir. 1988).

In establishing motivation, it has been consistently held that the requisite motivation to support the conclusion of obviousness is not an abstract concept, but must stem from the prior art as a whole to impel one having ordinary skill in the art to modify a reference or combine references with a reasonable expectation of successfully achieving some particular realistic

objective. See, for example, Interconnect Planning Corp. v. Feil, 227 F.2d 1132, 227 USPQ 543 (Fed. Cir. 1985).

1. Claim 3.

Dependent claim 3 recites receiving an un-deletion instruction regarding the file, and restoring an original name to the file without permanently deleting the renamed file.

Initially, claim 3 depends from claim 1. The disclosure of Manley et al. does not cure the deficiencies in the disclosure of Mattis et al. identified above with regard to claim 1. Claim 3 is, therefore, patentable over Mattis et al. and Manley et al., whether taken alone or in any reasonable combination, for at least the reasons given with regard to claim 1.

Additionally, neither Mattis et al. nor Manley et al., whether taken alone or in any reasonable combination, discloses or suggests the combination of features recited in claim 3. For example, neither Mattis et al. nor Manley et al. discloses or suggests restoring an original name to a renamed file without permanently deleting the renamed file.

The Examiner admitted that Mattis et al. does not disclose this feature, but alleged that Manley et al. discloses this feature and cited paragraph 0067 of Manley et al. for support. Final Office Action, page 7. Appellants disagree.

At paragraph 0067, Manley et al. discloses:

> FIG. 7 shows an exemplary inode file system structure 700 after a file data block has been modified. In this illustrative example, file data which is stored at disk block 520C is modified. The exemplary WAFL file system writes the modified contents to disk block 520C', which is a new location on disk. Because of this new location, the inode file data which is stored at disk block (515) is rewritten so that it points to block 520C'. This modification causes WAFL to allocate a new disk block (715) for the updated version of the data at 515. Similarly, the inode file indirect block 510 is rewritten to block 710 and direct block 512 is rewritten to block 712, to point to the newly revised inode 715. Thus, after a file data block has been modified the snapshot inode 605 contains a pointer to the original inode file system indirect block 510 which, in turn, contains a link to the inode 515. This inode 515 contains pointers to the original file data blocks 520A, 520B and

> 520C. However, the newly written inode 715 includes pointers to unmodified file data blocks 520A and 520B. The inode 715 also contains a pointer to the modified file data block 520C' representing the new arrangement of the active file system. A new file system root inode 705 is established representing the new structure 700. Note that metadata in any snapshotted blocks (e.g. blocks 510, 515 and 520C) protects these blocks from being recycled or overwritten until they are released from all snapshots. Thus, while the active file system root 705 points to new blocks 710, 712, 715 and 520C', the old blocks 510, 515 and 520C are retained until the snapshot is fully released.

In this section, Manley et al. discloses that after a file data block has been modified, the snapshot inode contains a pointer to the original inode file system indirect block which contains a link to the inode. Nowhere in this section, or elsewhere, does Manley et al. disclose or suggest restoring an original name to a renamed file without permanently deleting the renamed file, as required by claim 3.

The Examiner also alleged that because "there is a pointer to the original file then the original name must be re-assigned." Final Office Action, page 14. Appellants submit that the Examiner is not addressing the feature of claim 3. Claim 3 does not recite a pointer to an original file or reassigning an original name, but instead recites restoring an original name to a renamed file without permanently deleting the renamed file. The Examiner's allegation falls short of establishing a prima facie case of obviousness with regard to claim 3.

The Examiner also cited paragraph 0118 of Manley et al. for allegedly disclosing restoring an original name to a renamed file without permanently deleting the renamed file. Final Office Action, page 14. Appellants disagree.

At paragraph 0118, Manley et al. discloses:

> However, in the illustrative embodiment, if the source inodes received at the destination refer to inodes in the inode map 1400, then the directory stage creates (on the current built-up snapshot directory 1330) a file entry having the desired file name. This name can be exactly the name derived from the source. A hard link 1332 (i.e. a Unix-based link enables multiple names to be assigned to a discrete file) is created between that file on the snapshot directory 1330 and the entry in the purgatory directory. By so linking the entry,

> it is now pointed to by both the purgatory directory and the file on the snapshot directory itself. When the purgatory directory root is eventually deleted (thereby killing off purgatory) at the end of the data stream transfer, the hard link will remain to the entry, ensuring that the specific entry in the purgatory directory will not be deleted or recycled (given that the entry's link count is still greater than zero) and a path to the data from the file on the new directory is maintained. Every purgatory entry that eventually becomes associated with a file in the newly built tree will be similarly hard linked, and thereby survive deletion of the purgatory directory. Conversely, purgatory entries that are not relinked will not survive, and are effectively deleted permanently when purgatory is deleted.

In this section, Manley et al. discloses a purgatory entry that is hard linked will survive deletion of the purgatory directory and that a purgatory entry that is not relinked will not survive and will be deleted permanently. Nowhere in this section, or elsewhere, does Manley et al. disclose or suggest restoring an original name to a renamed file without permanently deleting the renamed file, as required by claim 3.

The Examiner alleged that this section of Manley et al.

> means that the file has been marked to be deleted was given a hard link to the original file, represented by the specific entry, so the path to the data remains. Once the file is un-deleted, the file survives deletion and is given its original name because of the original linkage. As a result, Mattis in view of Manley, disclose restoring an original name to a renamed file without permanently deleting the renamed file.

Final Office Action, page 14. As explained above, nowhere in paragraph 0118, or elsewhere, does Manley et al. disclose or suggest restoring an original name to a renamed file without permanently deleting the renamed file, as required by claim 3. The Examiner's allegations to the contrary find no support in the disclosure of Manley et al. and result solely from the use of impermissible hindsight reasoning.

In the Advisory Action, the Examiner cited paragraph 0132 of Manley et al. for allegedly disclosing restoring an original name to a renamed file without permanently deleting the renamed file. Advisory Action, page 2. Appellants disagree.

At paragraph 0132, Manley et al. discloses:

> One noted advantage to the rollback according to this embodiment is that it enables the undoing of set of changes to a replicated data set without the need to maintain separate logs or consuming significant system resources. Further the direction of rollback--past-to-present or present-to-past-is largely irrelevant. Furthermore, use of the purgatory directory, and not deleting files, enables the rollback to not affect existing NFS clients. Each NFS client accesses files by means of file handles, containing the inode number and generation of the file. If a system deletes and recreates a file, the file will have a different inode/generation tuple. As such, the NFS client will not be able to access the file without reloading it (it will see a message about a stale file handle). The purgatory directory, however, allows a delay in unlinking files until the end of the transfer. As such, a rollback as described above can resurrect files that have just been moved into purgatory, without the NFS clients taking notice.

In this section, Manley et al. discloses a rollback process that enables the undoing of a set of changes to a replicated data set. Even though Manley et al. discloses undoing a set of changes, Manley et al. does not disclose or suggest restoring an original name to a renamed file without permanently deleting the renamed file, as required by claim 3.

For at least these reasons, it is respectfully submitted that claim 3 is patentable over Mattis et al. and Manley et al., whether taken alone or in any reasonable combination, under 35 U.S.C. § 103. Reversal of the rejection of claim 3 is respectfully requested.

C. The Rejection Under 35 U.S.C. § 103(a) Based on Mattis et al. (U.S. Patent No. 6,209,003) in View of Hisgen et al. ("New-Value Logging in the Echo Replicated File System," June 23, 1993) Should be Reversed.

As explained above, the initial burden of establishing a prima facie basis to deny patentability to a claimed invention is always upon the Examiner. In re Oetiker, 977 F.2d 1443, 24 USPQ2d 1443 (Fed. Cir. 1992). In rejecting a claim under 35 U.S.C. § 103, the Examiner must provide a factual basis to support the conclusion of obviousness. In re Warner, 379 F.2d 1011, 154 USPQ 173 (CCPA 1967). Based upon the objective evidence of record, the Examiner

is required to make the factual inquiries mandated by Graham v. John Deere Co., 86 S.Ct. 684, 383 U.S. 1, 148 USPQ 459 (1966). The Examiner is also required to explain how and why one having ordinary skill in the art would have been led to modify an applied reference and/or combine applied references to arrive at the claimed invention. Uniroyal, Inc. v. Rudkin-Wiley Corp., 837 F.2d 1044, 5 USPQ2d 1434 (Fed. Cir. 1988).

In establishing motivation, it has been consistently held that the requisite motivation to support the conclusion of obviousness is not an abstract concept, but must stem from the prior art as a whole to impel one having ordinary skill in the art to modify a reference or combine references with a reasonable expectation of successfully achieving some particular realistic objective. See, for example, Interconnect Planning Corp. v. Feil, 227 F.2d 1132, 227 USPQ 543 (Fed. Cir. 1985).

1. Claims 8 and 9.

Dependent claim 8 recites identifying an orphaned chunk, including providing a mapping of file names to chunks, and identifying a chunk, as the orphaned chunk, that is not reachable from any of the file names; and deleting the orphaned chunk.

Claim 8 depends from claim 1. The disclosure of Hisgen et al. does not cure the deficiencies in the disclosure of Mattis et al. identified above with regard to claim 1. Claim 8 is, therefore, patentable over Mattis et al. and Hisgen et al., whether taken alone or in any reasonable combination, for at least the reasons given with regard to claim 8.

For at least these reasons, it is respectfully submitted that claims 8 and 9 are patentable over Mattis et al. and Hisgen et al., whether taken alone or in any reasonable combination, under 35 U.S.C. § 103. Reversal of the rejection of claims 8 and 9 is respectfully requested.

2. Claims 14, 15, 18, and 19.

Independent claim 14 is directed to a method for deleting orphaned chunks of a plurality of chunks stored by a plurality of servers. The method comprises providing a mapping of file names to chunks; identifying chunks, as orphaned chunks, that are not reachable from any of the file names; deleting the orphaned chunks; receiving, from the servers, information concerning chunks stored by the servers; and identifying, to one of the servers, one of the chunks that corresponds to one of the deleted orphaned chunks.

Neither Mattis et al. nor Hisgen et al., whether taken alone or in any reasonable combination, discloses or suggests the combination of features recited in claim 14. For example, neither Mattis et al. nor Hisgen et al. discloses or suggests identifying, to one of the servers, one of the chunks that corresponds to one of the deleted orphaned chunks.

The Examiner admitted that Mattis et al. does not disclose this feature, but alleged that Hisgen et al. discloses identifying, to the servers, ones of the chunks that are orphaned chunks and cited page 24, paragraph 3, lines 4-5, of Hisgen et al. for support. Final Office Action, page 9. Appellants disagree.

At page 24, paragraph 3, Hisgen et al. discloses:

> Orphan list. An orphan file is a file that is no longer in the name space, in that no entry in any directory points to it, but it is still open on some client machine and therefore must still be kept in existence. (We use the term "orphan" because an orphan file has no parent directory.) The orphan list lists all orphan files. When the Echo distributed client-server caching algorithm [21] reveals that nobody has an orphan file open, EchoBox can remove the file from its orphan list and delete the file, returning its pages to the disk space allocator and removing it from the fid map.

In this section, Hisgen et al. discloses that when an orphan file is not open, the file can be removed from the orphan list and deleted. Nowhere in this section, or elsewhere, does Hisgen et al. disclose or suggest identifying, to one of the servers from which information concerning

chunks stored by the servers was received, one of the chunks that corresponds to one of the deleted orphaned chunks, as required by claim 14.

For at least these reasons, it is respectfully submitted that claims 14, 15, 18, and 19 are patentable over Mattis et al. and Hisgen et al., whether taken alone or in any reasonable combination, under 35 U.S.C. § 103. Reversal of the rejection of claims 14, 15, 18, and 19 is respectfully requested.

3. Claim 16.

Dependent claim 16 recites deleting, by the one of the servers, the one of the chunks that corresponds to one of the orphaned chunks.

Initially, claim 16 depends from claim 14. Claim 16 is, therefore, patentable over Mattis et al. and Hisgen et al., whether taken alone or in any reasonable combination, for at least the reasons given with regard to claim 14.

Additionally, neither Mattis et al. nor Hisgen et al., whether taken alone or in any reasonable combination, discloses or suggests the combination of features recited in claim 16. The Examiner alleged that Hisgen et al. discloses this feature and cited page 24, paragraph 3, lines 5-8, of Hisgen et al. for support. Final Office Action, page 9. Appellants disagree.

Page 24, paragraph 3, of Hisgen et al. is reproduced above. In this section, Hisgen et al. discloses that when an orphan file is not open, the file can be removed from the orphan list and deleted. Nowhere in this section, or elsewhere, does Hisgen et al. disclose or suggest deleting, by the one of the servers, the one of the chunks that corresponds to one of the orphaned chunks, as required by claim 16.

Moreover, the Examiner has provided absolutely no motivation for combining this alleged feature of Hisgen et al. with the system of Mattis et al. Therefore, the Examiner has not established a proper rejection under 35 U.S.C. § 103.

For at least these reasons, it is respectfully submitted that claim 16 is patentable over Mattis et al. and Hisgen et al., whether taken alone or in any reasonable combination, under 35 U.S.C. § 103. Reversal of the rejection of claim 16 is respectfully requested.

4. Claim 17.

Dependent claim 17 recites that the deletion of the orphaned chunks occurs as part of a garbage collection process.

Initially, claim 17 depends from claim 14. Claim 17 is, therefore, patentable over Mattis et al. and Hisgen et al., whether taken alone or in any reasonable combination, for at least the reasons given with regard to claim 14.

Additionally, neither Mattis et al. nor Hisgen et al., whether taken alone or in any reasonable combination, discloses or suggests the combination of features recited in claim 17. The Examiner alleged that Hisgen et al. discloses the deletion of orphaned chunks and Mattis et al. discloses that the deletion of orphaned chunks occurs as part of a garbage collection process. Final Office Action, page 9. Appellants submit that the Examiner's rejection is improper.

The Examiner is dissecting claim features into parts and identifying portions of two separate references as allegedly disclosing these parts. Nowhere does Mattis et al. disclose or suggest orphaned chunks. Therefore, Mattis et al. cannot disclose or suggest deleting orphaned chunks as part of a garbage collection process.

In addition, even if Hisgen et al. discloses deleting orphaned chunks and Mattis et al. discloses garbage collection, the Examiner has provided no motivation for combining these alleged features of Hisgen et al. and Mattis et al. Therefore, the Examiner has not established a proper rejection under 35 U.S.C. § 103.

For at least these reasons, it is respectfully submitted that claim 17 is patentable over Mattis et al. and Hisgen et al., whether taken alone or in any reasonable combination, under 35 U.S.C. § 103. Reversal of the rejection of claim 17 is respectfully requested.

VIII. CONCLUSION

In view of the foregoing arguments, Appellants respectfully solicit the Honorable Board to reverse the Examiner's rejections of claims 1-25 under 35 U.S.C. §§ 102 and 103.

To the extent necessary, a petition for an extension of time under 37 C.F.R. § 1.136 is hereby made. Please charge any shortage in fees due in connection with the filing of this paper, including extension of time fees, to Deposit Account No. 50-1070 and please credit any excess fees to such deposit account.

Respectfully submitted,

HARRITY SNYDER, L.L.P.

/Paul A. Harrity/
Paul A. Harrity
Reg. No. 39,574

Date: December 13, 2006
11350 Random Hills Road
Suite 600
Fairfax, Virginia 22030
(571) 432-0800

CLAIM APPENDIX

1. A method for deleting one or more of a plurality of files, the files including one or more chunks stored by a plurality of servers, the method comprising:

identifying a file to be deleted;

renaming the identified file;

permanently deleting the renamed file a predetermined amount of time after renaming the identified file as part of a garbage collection process;

receiving, from the servers, information concerning chunks stored by the servers; and

identifying, to one of the servers, one of the chunks that corresponds to the permanently deleted file.

2. The method of claim 1, wherein the identifying a file to be deleted includes:

receiving a deletion instruction regarding the file.

3. The method of claim 2, further comprising:

receiving an un-deletion instruction regarding the file; and

restoring an original name to the file without permanently deleting the renamed file.

4. The method of claim 1, wherein the predetermined amount of time is a user-configurable amount of time.

5. The method of claim 4, wherein the user-configurable amount of time differs for

different ones of the files.

6. The method of claim 1, wherein metadata is associated with the files; and

wherein the permanently deleting the renamed file includes erasing the metadata associated with the renamed file.

7. The method of claim 1, further comprising:

deleting, by the one of the servers, the one of the chunks that corresponds to the permanently deleted file.

8. The method of claim 1, further comprising:

identifying an orphaned chunk, including:

providing a mapping of file names to chunks, and

identifying a chunk, as the orphaned chunk, that is not reachable from any of the file names; and

deleting the orphaned chunk.

9. The method of claim 8, wherein metadata is associated with the chunks; and

wherein the deleting the orphaned chunk includes erasing the metadata associated with the orphaned chunk.

10. The method of claim 1, further comprising:

maintaining versions of the chunks;

identifying a stale chunk based on the versions of the chunks; and

deleting the stale chunk.

11. The method of claim 10, wherein metadata is associated with the chunks; and

wherein the deleting the stale chunk includes erasing the metadata associated with the stale chunk.

12. A system for deleting a file that includes data stored by a plurality of servers, comprising:

means for identifying a file to be deleted;

means for logging deletion of the identified file;

means for permanently deleting the file during a garbage collection process that occurs after logging deletion of the identified file;

means for receiving, from the servers, information concerning data stored by the servers; and

means for identifying, to one of the servers, that of the data that corresponds to the file that was permanently deleted.

13. A file system, comprising:

a plurality of servers configured to store files as chunks, each of the files including one or more of the chunks; and

a master connected to the servers and configured to:

identify one of the files to be deleted,

rename the identified file,

permanently delete one or more chunks associated with the renamed file a predetermined amount of time after renaming the identified file as part of a garbage collection process,

receive, from the servers, information concerning chunks stored by the servers, and

identify, to one of the servers, one of the chunks that corresponds to one of the one or more permanently deleted chunks.

14. A method for deleting orphaned chunks of a plurality of chunks stored by a plurality of servers, the method comprising:

providing a mapping of file names to chunks;

identifying chunks, as orphaned chunks, that are not reachable from any of the file names;

deleting the orphaned chunks;

receiving, from the servers, information concerning chunks stored by the servers; and

identifying, to one of the servers, one of the chunks that corresponds to one of the deleted orphaned chunks.

15. The method of claim 14, wherein metadata is associated with the chunks; and

wherein the deleting the orphaned chunks includes erasing the metadata associated with the orphaned chunks.

16. The method of claim 14, further comprising:

deleting, by the one of the servers, the one of the chunks that corresponds to one of the orphaned chunks.

17. The method of claim 14, wherein the deletion of the orphaned chunks occurs as part of a garbage collection process.

18. A system for deleting orphaned chunks of a plurality of chunks stored by a plurality of servers, comprising:

means for mapping file names to chunks;

means for identifying chunks, as orphaned chunks, that are not reachable from any of the file names;

means for deleting the orphaned chunks as part of a garbage collection process;

means for receiving, from the servers, information concerning chunks stored by the servers; and

means for identifying, to one of the servers, one of the chunks that corresponds to one of the deleted orphaned chunks.

19. A file system, comprising:

a plurality of servers configured to store files as chunks, each of the files including one or more of the chunks; and

a master connected to the servers and configured to:

map file names to chunks,

identify chunks, as orphaned chunks, that are not reachable from any of the file names,

delete the orphaned chunks,

receive, from the servers, information concerning chunks stored by the servers, and

identify, to one of the servers, one of the chunks that corresponds to one of the deleted orphaned chunks.

20. A method for deleting stale replicas of chunks, the replicas being stored by a plurality of servers, the method comprising:

associating version information with replicas of chunks;

identifying stale replicas based on the associated version information;

deleting the stale replicas;

receiving, from the servers, information concerning replicas stored by the servers; and

identifying, to one of the servers, one of the replicas that corresponds to one of the deleted stale replicas.

21. The method of claim 20, wherein the version information for one of the replicas is

updated each time a lease is granted for the one of the replicas.

22. The method of claim 20, further comprising:

deleting, by the one of the servers, the one of the replicas that corresponds to one of the stale replicas.

23. The method of claim 20, wherein the deletion of the stale replicas occurs as part of a garbage collection process.

24. A system for deleting stale replicas of chunks, the replicas being stored by a plurality of servers, the system comprising:

means for generating version information for replicas of chunks;

means for identifying stale replicas based on the generated version information;

means for deleting the stale replicas as part of a garbage collection process;

means for receiving, from the servers, information concerning replicas stored by the servers; and

means for identifying, to one of the servers, one of the replicas that corresponds to one of the deleted stale replicas.

25. A file system that stores files as chunks, comprising:

a plurality of servers configured to store files as chunks; and

a master connected to the servers and configured to:

associate version information with the chunks,

identify stale chunks based on the associated version information,

delete the stale chunks,

receive, from the servers, information concerning replicas stored by the servers, and

identify, to one of the servers, one of the replicas that corresponds to one of the deleted stale chunks.

EVIDENCE APPENDIX

None

RELATED PROCEEDINGS APPENDIX

None